Generation of Controlled Analog Emissions from Embedded Devices using Software Stress Methods

Oren Sternberg, Jonathan H. Nelson, Israel Perez, Kristopher Buchanan, Sara Wheeland and John D. Rockway

Space and Naval Warfare Systems Center Pacific (SSC Pacific), San Diego, CA 92152-5001

Abstract: *In this paper, we present a new method that uses software diagnostic tools to study the generation of induced spurious physical emissions from embedded devices over air-gapped (remote) channels. With this methodology, spurious emissions are induced during controlled computer operations such as dynamic memory allocation, hard disk writing and computations. Each stressing operation creates a pulse in an amplitude shift keying scheme. These software techniques can provide repeatable measurements of embedded devices for mapping unwanted emissions over air-gapped channels.*

Keywords: air-gap; gnu radio; modem; sensors; software-defined radio; spurious emission

Introduction

Communication using induced spurious physical phenomena from computers secured via an air gap has been demonstrated recently by harnessing both thermal signals and electromagnetic signals [1,2]. Traditionally, covert channel techniques focused on networked systems and are widely discussed in the literature [1-5]. Other offensive sideways channel techniques which focused on network-induced physical phenomena to fingerprint a computer of interest have also been demonstrated [6]. Additional techniques using power differentials, side channel attacks, or statistical power analysis retrieving information on computers and embedded devices are widely discussed in the literature [7-13].

As many embedded devices are now linked with software control and diagnostics, defense against new types of attacks warrants an understanding of unwanted signal generation. We examine this connection by observing the emission profile of an embedded device while executing specific software commands.

The goal of this paper is to present a new methodology [14] that uses software diagnostic tools to control the generation of unwanted physical emissions [15,16] from embedded devices over air-gapped channels. Such emissions are induced via software stress testing and diagnostic and security applications including StressLinux (Linux) [17], KALI (Linux) [18] and a multitude of tools in Windows [19]. Primarily, these tools monitor and address load, stability, and environmental controls for personal computers as well as mobile and embedded devices. Here, we repurpose these tools to stress embedded devices, which results in the occurrence of new induced emissions. Further, these emissions are repeatable and appear around the normal, background response of the device. We present three different software stress techniques that induce new and unwanted emissions over the background response of a given device. To demonstrate the control available with this methodology, a final software stress technique is devised to induce a repeatable sequence of controlled emissions over the embedded device. This novel method can be used to calibrate the emission profile of a device, which can help later in differentiating between normal operations and those operations from an unwanted attack.

Analog Enablers (Hardware)

Spurious signals, also known as unwanted, unintended, or out-of-band emissions, are regulated by over 119 countries without a concise definition [15-16]. These emissions are inherent to powered electronic components, subsystems and systems that might cause undesired interference. As an example, Fig. 1 presents the measured inherent broadband Radio Frequency (RF) response from an embedded device, an Intel Galileo 2 (Quark SoC X1000). Fig.1 shows the emission spectrum response from the Intel Galileo 2 measured with a Tektronix Real time Spectrum Analyzer (RTSA) 3408B [14]. We observed two strong emission bands from the Intel Galileo 2. One band from 385 MHz to 387 MHz is observed for a duration of 7.2 seconds. An additional band is observed from 397 to 400 MHz. These responses range in magnitude from -58 to -54 dBm.

Digital Enablers (Software)

Software diagnostic tools monitor and address computer load, stability, and environmental control across a network. Fig. 2 demonstrates how these tools control and measure stress of Central Processing Units (CPUs). In this test, the command [20] was executed on a MacBook Pro (Darwin Kernel 14.05) to stress the CPU load.

```
% stress –cpu 500 –t5
```

The stress code employs 500 CPU workers (e.g., square root operations) for a duration of 5 seconds. A built-in activity monitor reads and records the CPU load. As expected, the CPU load spike is clearly observed under this stress operation.

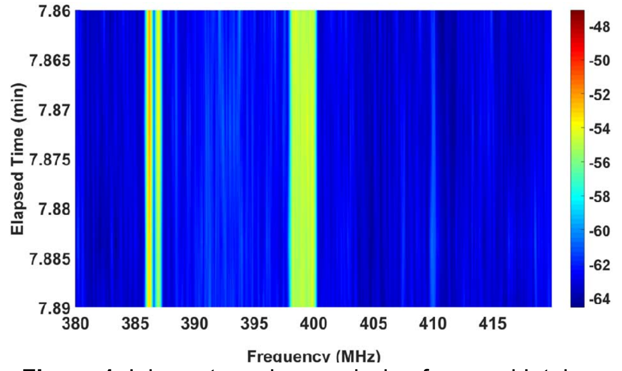
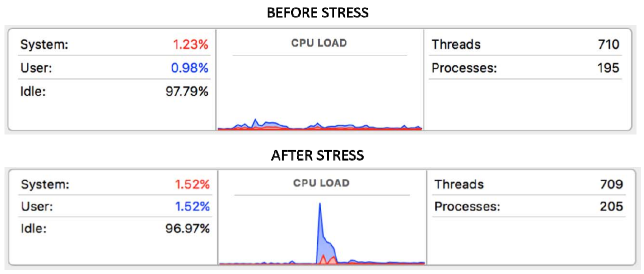


Figure 1: Inherent spurious emission from and Intel Galileo 2, scale in dBm



BEFORE STRESS

		CPU LOAD		
System:	1.23%		Threads	710
User:	0.98%		Processes:	195
Idle:	97.79%			

AFTER STRESS

		CPU LOAD		
System:	1.52%		Threads	709
User:	1.52%		Processes:	205
Idle:	96.97%			

Figure 2: CPU performance of a Darwin kernel before (top) and after (bottom) harmonic stressing using command % stress –cpu 500 –t5

To repurpose this type of stress technique for an embedded computer, we used the wrapper diagnostic tool, "workload generator for POSIX systems" [20]. This tool allows the user to control the stress across memory, file and CPU operations. We then performed a similar CPU operation but on Portable Operating System Interface (POSIX) embedded devices running Linux kernels. Fig. 3 presents measurements for the above stress command, *% stress –cpu 500 –t5*, executed on the aforementioned Intel Galileo 2.

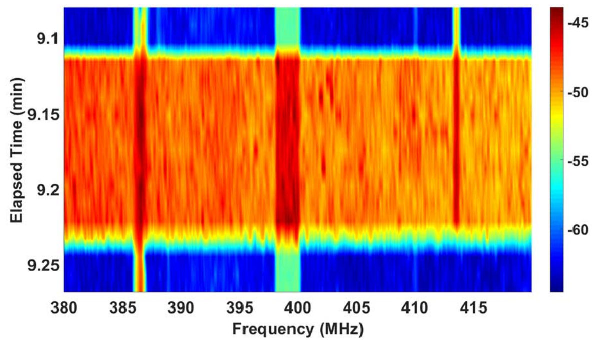


Figure 3: Induced emission from Intel Galileo 2, scale in dBm

The results indicates the stress *–cpu* flag ramping the CPU, inducing new responses around the clock frequencies of these devices, see Fig 3. A wider emission response was observed ranging from 380 MHz to 420 MHz for a duration of 7.2 seconds. The previous two bands of 385 MHz to 387 MHz and 397 MHz to 400 MHz were also observed to have an increase in peak magnitude from -54dBm to -48dBm.

Emission Inducing Stress Loading

In this section, a series of experiments are now presented for three different stress loading techniques executed on a Beagle Bone Rev C device (ARM Cortex-A8 processor). The resulting induced emissions are characterized and the different techniques are compared.

Stress Loading Scripts

Each of the following three scripts employs a digital enabler technique to trigger emissions by brute force on an embedded Beagle Bone Black Rev C device running a Linux kernel.

(1) *File operation, hard drive (HDD) [21]* –The *write()* function writes bytes from the buffer to the file referenced by the file descriptor (). The *unlink()* function deletes a name from the file system. We induce stress using the following execution code:

% stress -d 1 –timeout 5s (HDD stress)

(2) *Memory allocation [21]* – The malloc()/free() function allocates size in bytes and returns a pointer to the allocated memory. If the size is 0, then *malloc()* returns either NULL or a unique pointer value that can be passed to *free()*. The *free()* function frees the memory space indicated by a pointer, i.e. what is returned by *malloc()*. We execute this command using the following script:

% stress – m 1 –timeout 5s (Memory stress)

(3) *CPU operation [21]* – The *sqrt()* function simply returns the nonnegative square root of x. The compiled stress code performs 500 *sqrt()* operations for 5s with the following command:

% stress – cpu 500 –timeout 5s (CPU stress)

Experimental Setup

The experimental setup for the measurements uses a 150A EMC Probe Amplifier and EMC probe that is positioned near the device to collect the emissions. A USRP N210 software-defined radio receives the data. Table 1 lists the hardware and software components used in the series of experiments.

Table 1: Hardware and Software Tools

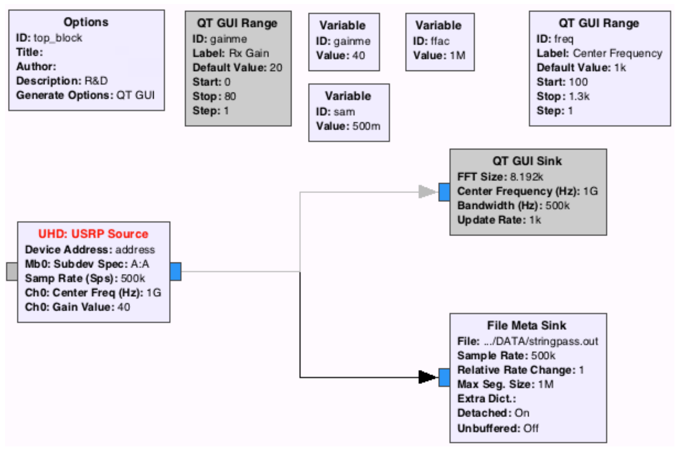
HARDWARE	
Device	**Model**
Software Defined Radio	USRP N210
EMC Probes	100 Series
EMC Probe amplifier	105A
Development Platform	Beagle Bone Black Rev C
MacBook Pro	2015
Desktop	HP Xeon
SOFTWARE (embedded device)	
Stress[20], CRON (job scheduler), python, c, bash	
SOFTWARE (receiver end)	
GNU RADIO [21] , c, python, OCTAVE, MATLAB	

The software defined radio source and probes used the following parameters for the tests: Center Frequency Fc = 1 GHz, Bandwidth BW = 1 MHz, Sample Rate SR = 500,000 Samples/Second, FFT Size = 8192, and probe distance d = 2 cm. Fig. 4 presents the flow diagram for the data collection procedure coded in GNU Radio. Where in the GNU radio interface the USRP N210 acts as a source for transferring the recorded data into a binary file for post processing (File Sink).

For each experiment, we monitor and record a 1 MHz band signal with the probe as a function of the three stress types.

Experimental Results

The following plots present the emission measurements from the Beagle Bone Black Rev C. using the three different stress scripts. In all three cases, we observed the emission from the clock frequency of the device at $f_{Clock} =$ 1.0000 GHz and a controlled induced emission behavior at $f_I =$ 1.0000367 GHz (Figs. 5-6).


Options
ID: top_block
Title:
Author:
Description: R&D
Generate Options: QT GUI

QT GUI Range
ID: gainme
Label: Rx Gain
Default Value: 20
Start: 0
Stop: 80
Step: 1

Variable
ID: gainme
Value: 40

Variable
ID: ffac
Value: 1M

Variable
ID: sam
Value: 500m

QT GUI Range
ID: freq
Label: Center Frequency
Default Value: 1k
Start: 100
Stop: 1.3k
Step: 1

QT GUI Sink
FFT Size: 8.192k
Center Frequency (Hz): 1G
Bandwidth (Hz): 500k
Update Rate: 1k

UHD: USRP Source
Device Address: address
Mb0: Subdev Spec: A:A
Samp Rate (Sps): 500k
Ch0: Center Freq (Hz): 1G
Ch0: Gain Value: 40

File Meta Sink
File: .../DATA/stringpass.out
Sample Rate: 500k
Relative Rate Change: 1
Max Seg. Size: 1M
Extra Dict.:
Detached: On
Unbuffered: Off

Figure 4: Graphical illustration of collection using GNU RADIO [21]

Fig. 6 presents the RF response of the Beagle Bone Black under the CPU operation stress script. Fig. 6 (top) shows the normal emission spectrum response from the Beagle Bone Black Rev C measured with a Tektronix RTSA 3408B [14]. We observed an induced emission at 1.00000 GHz (bandwidth ~30 MHz). The magnitude of the peak is -76 dBm and the noise floor is -88 dBm. Fig. 6 (bottom) shows the response of the device under the CPU operation stress script. The induced stress emission is clearly visible at a magnitude of -83 dBm. The significant rise in the noise envelope in the stressed case is characteristic of the CPU workload technique.

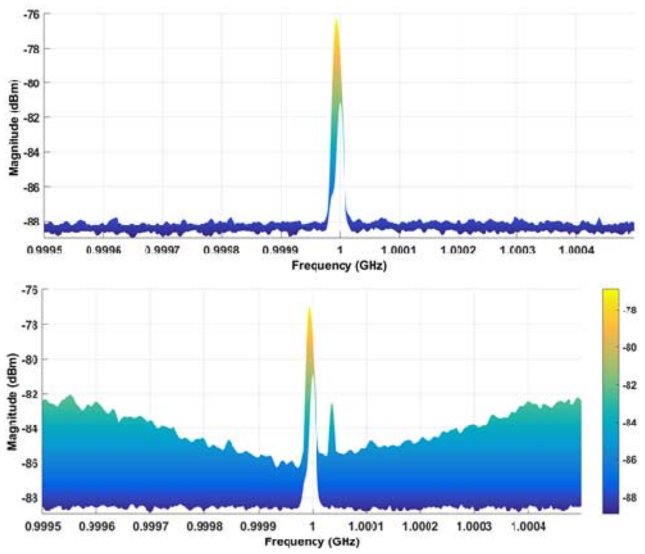


Figure 5: Background response from Beagle Bone (top) and CPU stress-induced response showing new peak and a rise in the noise envelope (bottom), scale in dBm

Fig. 6 compares the measurements for the three stress techniques in a spectrum plot. Each technique has a unique signal response and induced noise envelope. The memory allocation stress process has the longest RF response, Fig 6(middle) (~8 seconds), while the HDD stress has the shortest (~3 seconds). The RF emission from memory allocation exhibits greater control with induced emission; however the HDD process, Fig. 7 (top) has a higher signal to noise ratio (SNR).

The SNR is defined as the mean power at the peak tone frequency f_r = 1.000367 GHz divided by the mean noise spectrum.

$$SNR = \frac{\overline{S}(F_r)}{\overline{N}_S} \tag{1}$$

The SNR for HDD stress, CPU stress and dynamic memory allocation was experimentally calculated as 18.336, 34.744, and 30.68 dB, respectively.

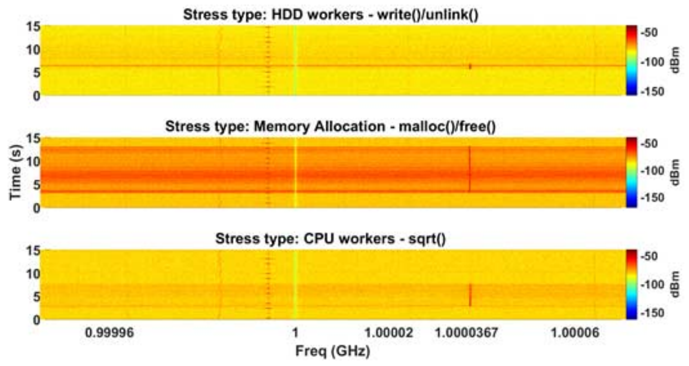
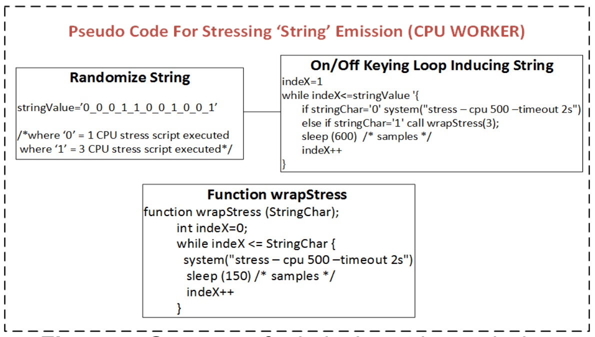


Figure 6: RF response using HDD (top), memory allocation (middle), and CPU (bottom) stress operations, scale in dBm

The *malloc()* memory allocation process speed suggests that the mean noise level is higher than the CPU and HDD process (~ -80dBm vs ~ -85dBm (i.e. faster to access memory than disk)). However the SNR values at peak tone are higher by ~12dB and ~15dB for HDD and CPU respectively, making CPU and HDD stress better candidates at the peak tone frequency. In the next section we employ the CPU stress process to demonstrate an experiment that repeatedly induces the desired unwanted emission.

Controlled Stress Loading Emission Experiment

To investigate the repeatability and control of the software bad induced emissions, a sequence of stress scripts is executed as a binary string, see Fig. 7 for the process. A binary string of value ‘00011001001’ is encoded (Fig. 8 – Randomize String) using CPU stress techniques and sent over the embedded system with a controlled spurious emission for reconstruction. A while-loop pushes 0’s and 1’s with a 750 sample spacing between digits (Fig 8 – Function WrapStress). For these tests, a ‘0’ string equals one event and the corresponding induced emission is observed over 150 samples with a 750 sample spacing. A ‘1’ string includes three events and the corresponding induced emission is observed over 150 samples and a spacing of < 200 samples between events. A reconstruction of the string is shown in Fig. 8 as waterfall (top) and logic plots (bottom).


Pseudo Code For Stressing ‘String’ Emission (CPU WORKER)
Randomize String
On/Off Keying Loop Inducing String
Function wrapStress

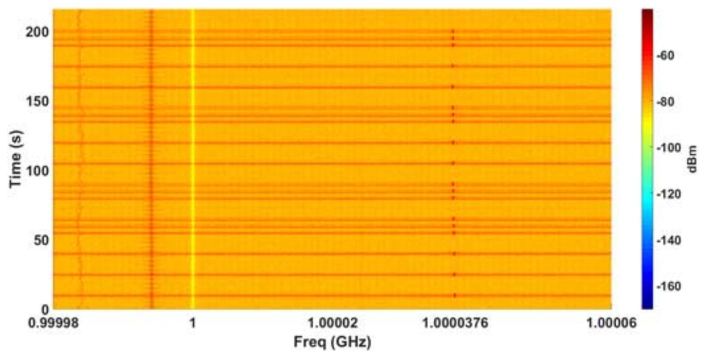
```
stringValue='0_0_0_1_1_0_0_1_0_0_1'

/*where '0' = 1 CPU stress script executed
 where '1' = 3 CPU stress script executed*/
```

```
indeX=1
while indeX<=stringValue '{
    if stringChar='0' system("stress – cpu 500 –timeout 2s")
    else if stringChar='1' call wrapStress(3);
    sleep (600)  /* samples */
    indeX++
}
```

```
function wrapStress (StringChar);
        int indeX=0;
        while indeX <= StringChar {
          system("stress – cpu 500 –timeout 2s")
           sleep (150) /* samples */
           indeX++
        }
```

Figure 7: Sequence for inducing string emission



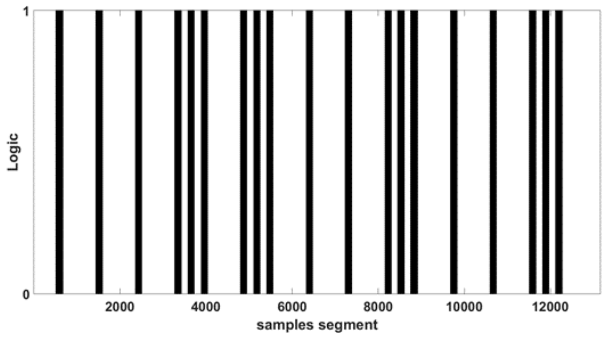


Figure 8: Reconstruction of the ‘00011001001’ in waterfall (top) and logic (bottom) plots transmitted using square root operations

Conclusion

This paper presented a method to induce RF spurious emissions over an embedded device using software stress techniques. The transmitting POSIX board does not require additional hardware for transmission. The receiver employs a probe and a software-defined radio to record generated emissions in a very short range of ~2 cm with a few milliseconds per byte using the *malloc()* stress process.

While our effort focuses on a new methodology to generate unwanted emissions, which could be used as a diagnostic tool, the question of protection against undesired data retrieval arises. Requirements for the control of electromagnetic interference characteristics of subsystems and equipment, MIL-STD-461, MIL-HDBK-1195 and others are mostly sufficient in shielding and characterizing unwanted spurious emissions. However, there is a gap in understanding the linkage between software operations and unwanted emission, which this paper sought to address. Future work will focus on other characterization techniques such as floating point, integer, bit manipulation, cache access and control flow for single and multi-core systems, in addition to finding other techniques to correlate power consumption with function execution.

References

[1] M. Guri, G. Kedma, A. Kachlon and Y. Elovici, "AirHopper: Bridging the Air-Gap between Isolated Networks and Mobile Phones using Radio Frequencies," in *9th IEEE International Conference on Malicious and Unwanted Software (MALCON 2014)*, Puero Rico, Fajardo, 2014.

[2] Guri, Mordechai, et al. "BitWhisper: Covert Signaling Channel between Air-Gapped Computers using Thermal Manipulations." arXiv preprint arXiv:1503.07919 (2015).

[3] Z. Z. X. a. H. W. Wu, "Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud," *in USENIX Security symposium*, 2012.

[4] G. V. Jie Chen, "CC-Hunter: Uncovering Covert Timing Channels on Shared Processor Hardware," in *MICRO-47 Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, 2014.

[5] H. W. a. H. W. Ki Suh Lee, "PHY Covert Channels: Can you see the Idles?," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI '14)* , Seattle, 2014.

[6] Kohno, Tadayoshi, Andre Broido, and Kimberly C. Claffy. *"Remote physical device fingerprinting." IEEE Transactions on Dependable and Secure Computing 2.2 (2005): 93-108.*

[7] *Murdoch, "Hot or Not: Revealing hidden services by Their Clock Skew", http://www.clc.cam.ac.uk/~sjm217*

[8] Kocher, C., Jaffe, J., Jun, B., Differential Power Analysis, Advances in Cryptology-CRYPTO' 99, LNCS1666, (1999), 388–397. Iokibe, Kengo, Tetsuo Amano, and Yoshitaka Toyota. "On-board decoupling of cryptographic FPGA to improve tolerance to side-channel attacks*." Electromagnetic Compatibility (EMC), 2011 IEEE International Symposium* on. IEEE, 2011.

[9] Lawson, Nate. "Side-channel attacks on cryptographic software." *Security & Privacy, IEEE* 7.6 (2009): 65-68.

[10] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," Cryptography Research, 1995; www.cryptography.com/resources/ whitepapers/TimingAttacks.pdf

[11] D. Brumley and D. Boneh, "Remote Timing Attacks Are Practical," *Proc. 12th Conf. Usenix Security Symp., Usenix Assoc*., 2003, p. 1.

[12] Brier, Eric, Christophe Clavier, and Francis Olivier. "Correlation power analysis with a leakage model." *Cryptographic Hardware and Embedded Systems*-CHES 2004. Springer Berlin Heidelberg, 2004. 16-29.

[13] Clavier, Christophe, Jean-Sébastien Coron, and Nora Dabbous. "Differential power analysis in the presence of hardware countermeasures." *Cryptographic Hardware and Embedded Systems*—CHES 2000. Springer Berlin Heidelberg, 2000.

[14] Patent Pending U.S. NAVY CASE# 103418 "Method of Wireless Transferring Data by Driving and Controlling Electronic Components Power Emissions," 2/2015.

[15] https://www.fcc.gov/encyclopedia/rules-regulations-title-47 (Title 47, Chapter I, Subchapter A-H , Part 15)

[16] https://transition.fcc.gov/mb/audio/bickel/world-govt-telecom.html worldwide government telecommunication sites

[17] http://www.stresslinux.org/sl/

[18] https://www.kali.org/

[19] https://www.raymond.cc/blog/test-system-stability-by-putting-heavy-load-on-system-resources

[20] http://people.seas.harvard.edu/~apw/stress/stress-1.0.4.tar.gz

[21] http://man7.org/linux/man-pages/man3/malloc.3.html

[22] Monti, Giuseppina, et al. "Resonant energy scavenger for sensor powering by spurious emissions from compact fluorescent lamps*." Sensors Journal, IEEE* 14.7 (2014): 2347-2354